

**OPERABILITY ASSESSMENT SYSTEM (OASYS)  
FINAL TECHNICAL REPORT**

**Howard Briscoe  
Jean MacMillan  
William Ferguson  
Richard W. Pew**

**BBN SYSTEMS AND TECHNOLOGIES  
10 Moulton Street  
Cambridge, Massachusetts 01238**

**HUMAN RESOURCES DIRECTORATE  
LOGISTICS RESEARCH DIVISION  
2698 G Street  
Wright-Patterson Air Force Base, Ohio 45433-7604**

**JULY 1996**

**FINAL TECHNICAL PAPER FOR PERIOD APRIL 1992 TO MARCH 1996**

**19970103 008**

**DTIC QUALITY INSPECTED 2**

**Approved for public release; distribution is unlimited**

**AIR FORCE MATERIEL COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573**

**ARMSTRONG  
LABORATORY**


## NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation, or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has revised this report and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nations.

This report has been reviewed and is approved for publication.

  
KURT BOLIN, Capt, USAF  
Program Manager

  
BERTRAM W. CREAM, GM-15, DAF  
Chief, Logistics Research Division

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE July 1996	3. REPORT TYPE AND DATES COVERED Final - March 1992 to March 1996	
4. TITLE AND SUBTITLE Operability Assessment System (OASYS) Final Technical Report			5. FUNDING NUMBERS  C - F33615-91-C-0012 PE - 63106F PR - 2940 TA - 03 WU - 02	
6. AUTHOR(S) Howard Briscoe      Richard W. Pew Jean MacMillan William Ferguson				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  BBN Systems and Technologies 10 Moulton Street Cambridge, Massachusetts 01238			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Armstrong Laboratory Human Resources Directorate Logistics Research Division 2698 G Street Wright-Patterson AFB, OH 45433-7604			10. SPONSORING / MONITORING AGENCY REPORT NUMBER  AL/HR-TR-1996- 0109	
11. SUPPLEMENTARY NOTES  Armstrong Laboratory Technical Monitor: Kurt Bolin, Capt, USAF, AL/HRGA, DSN 785-7773				
12a. DISTRIBUTION / AVAILABILITY STATEMENT  Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This document is the final report of the Operability Assessment system (OASYS) contract. The primary goals of the OASYS project were: 1) to develop a methodology for design, valuation, and maintenance of system operability throughout the life cycle of a system, and 2) to develop a system of computer tools to support that methodology. This document describes the operability assessment methodology and software products developed under the program. The report also details the design history of the software tools, the software test process, and lessons learned.				
14. SUBJECT TERMS Operability Human-in-the-loop simulation Human performance modeling Object-oriented simulation			15. NUMBER OF PAGES 55	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT SAR	

## GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to *stay within the lines* to meet *optical scanning requirements*.

**Block 1. Agency Use Only (Leave blank).**

**Block 2. Report Date.** Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

**Block 3. Type of Report and Dates Covered.** State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

**Block 4. Title and Subtitle.** A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

**Block 5. Funding Numbers.** To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

<b>C</b> - Contract	<b>PR</b> - Project
<b>G</b> - Grant	<b>TA</b> - Task
<b>PE</b> - Program Element	<b>WU</b> - Work Unit Accession No.

**Block 6. Author(s).** Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

**Block 7. Performing Organization Name(s) and Address(es).** Self-explanatory.

**Block 8. Performing Organization Report Number.** Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

**Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es).** Self-explanatory.

**Block 10. Sponsoring/Monitoring Agency Report Number.** (If known)

**Block 11. Supplementary Notes.** Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

**Block 12a. Distribution/Availability Statement.** Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

**DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."

**DOE** - See authorities.

**NASA** - See Handbook NHB 2200.2.

**NTIS** - Leave blank.

**Block 12b. Distribution Code.**

**DOD** - Leave blank.

**DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.

**NASA** - Leave blank.

**NTIS** - Leave blank.

**Block 13. Abstract.** Include a brief (*Maximum 200 words*) factual summary of the most significant information contained in the report.

**Block 14. Subject Terms.** Keywords or phrases identifying major subjects in the report.

**Block 15. Number of Pages.** Enter the total number of pages.

**Block 16. Price Code.** Enter appropriate price code (*NTIS only*).

**Blocks 17. - 19. Security Classifications.** Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

**Block 20. Limitation of Abstract.** This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

## PREFACE

The goals of the Operability Assessment System (OASYS) program were to develop a methodology for design, evaluation, and maintenance of system operability throughout the life cycle of a system and to develop a system of computer tools to support that methodology. OASYS provides the capability to quickly construct a system interface and simulation and to conduct human-in-the-loop experiments to assess system operability. Early consideration of these issues should result in improved system performance through better design and lower system cost through earlier identification of design issues. This effort was conducted by BBN Systems & Technologies in Cambridge, Massachusetts. This research was supported by the Logistics Research Division, Human Resources Directorate, Armstrong Laboratory, under Work Unit 2940-03-02.

# Table of Contents

	Page
1. Introduction.....	1
2. Relevant Documents .....	1
3. Project Overview .....	2
4. OASYS Prototype Test and Evaluation.....	4
5. OASYS Capabilities—A User's Perspective .....	5
5.1 Overview of OASYS Capabilities.....	6
5.2 Using OASYS to Resolve Operability Issues.....	7
5.2.1 The Challenge of Operability Assessment .....	7
5.2.2 Overview of OASYS Use.....	11
5.2.3 Developing Function and Task Decompositions.....	12
5.2.4 Specifying Task Flows and Task Allocations .....	15
5.2.5 Analyzing Task Timing and Operator Loads .....	18
5.2.6 Specifying Information Requirements.....	21
5.2.7 Selecting an Area for Prototyping .....	22
5.2.8 Developing Display Designs .....	22
5.2.9 Building and Using Dynamic Prototypes in Demonstrations and Experiments.....	23
6. OASYS System.....	28
6.1 System Overview.....	28
6.2 Documents .....	29
6.3 Task Analysis, Timing, and Load.....	30
6.4 Experiment System.....	32
6.4.1 Interface Design.....	33
6.4.2 Simulation Design .....	34
6.4.3 Experiment Design .....	36
6.4.4 Experiment Execution .....	38
6.4.5 Data Collection and Analysis .....	39

6.4.6	Human Performance Model Interface.....	40
6.4.7	Demonstration .....	41
7.	OASYS Software Testing .....	42
7.1.	Test Coverage .....	42
7.2	Test Process .....	43
8.	Lessons Learned.....	43
9.	In Retrospect .....	44
10.	References.....	46

**Figures**

1.	Using OASYS for Operability Assessment .....	11
2.	Task Analysis Tools.....	31
3.	OASYS-ES Experiment System .....	33

## **1. Introduction**

This document is the final report of the Operability Assessment System (OASYS) contract. The primary goals of the OASYS project were: 1) to develop a methodology for design, evaluation, and maintenance of system operability throughout the life cycle of a system, and 2) to develop a system of computer tools to support that methodology. A secondary goal for OASYS was to provide a testbed for testing and validation of Human Performance Models (HPM).

In this document we start with a summary of the major milestones of the project in Section 3. Throughout the project there was an attempt to provide continuing test and evaluation of the design and implementation of the tools using hands on testing by both Armstrong Laboratory and BBN human factors experts and by holding frequent design reviews attended by a review board consisting of potential users of the OASYS system. These evaluation activities are summarized in Section 4. Section 5 provides a description of the methodology the OASYS software tools were designed to support and an overview of the software delivered under this project. The design history for each of the tools, including the basis for technical decisions in the design and the lessons learned about the design process in retrospect, are described in Section 6. Section 7 describes the formal software test process. Finally, Sections 8 and 9 discuss some lessons learned in the process of using a rapid prototype development approach and comment on the possible future path of operability support and human performance modeling.

## **2. Relevant Documents**

The following documents were produced during the OASYS project to describe the OASYS system and how to use it:

- "*Concept of Operations*", CDRL A010, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, September 1993.
- "*System Requirements Specification*", CDRL A008, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, November 1994.
- "*User Manual*", CDRL A015, Contract F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, June, 1995.
- "*OASYS Programmers Manual*", CDRL A014, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, June 1995.



- *"Installation Guide"* (OASYS Task Analysis System (OASYS-TAS), CDRL A004, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, June 1995.

### **3. Project Overview**

In September of 1991 BBN submitted proposal number P92-LABS-C-042 in response to Armstrong Laboratory request for proposal number F33615-91-R-0012 for development of an Operability Assessment System for Integrated Simultaneous Engineering (OASIS). After proposals were evaluated, BBN was issued a contract for development of the OASIS (later changed to OASYS) system in February of 1992. The contract called for a four-phase effort over a 44 month period. The four phases consisted of:

- Phase I: System Requirements Analysis and Preliminary Architecture Definition.
- Phase II: System Prototype Design Specification.
- Phase III: Prototype Development.
- Phase IV: Demonstration, Validation, and Documentation of the Prototype.

The major events and milestones in each of these phases are summarized below.

#### **Phase I**

- The kickoff meeting for Phase I was held on April 21, 1992 at Wright Patterson Air Force Base (WPAFB).
- The OASIS Program Plan (CDRL A001) was submitted in May of 1992, and was reviewed as part of a Technical Status Review in June.
- The OASIS System Segment Specification (CDRL A006) was submitted in November of 1992.
- Phase I was completed in December of 1992 with the Software Requirements Review (SRR) held at BBN in Cambridge, Massachusetts.

#### **Phase II**

- Following the SRR, BBN received approval to proceed with Phase II.
- The first formal technical meeting in Phase II was held at WPAFB on February 11, 1993.
- Preliminary versions of the OASIS SRS (CDRL A009), and the IRS (CDRL A008) were delivered to AL and discussed in the Software System Review held at BBN on June 28, 1993.
- The Concept of Operation (CDRL A010) for OASYS was delivered in September.

- A detail system design document combining the Software Design Document (CDRL A011) and the Interface Design Document (CDRL A012) was delivered in November.
- Finally, the Critical Design Review was held at BBN on November 17 and 18, 1993.

### **Phase III**

- The Phase III kickoff meeting was held at WPAFB on February 2, 1994.
- Build 1 of the OASYS software was completed and installed at WPAFB at the end of June. A user manual for build 1 was prepared and a brief training session was conducted at that time.
- A second version of build 1 incorporating bug fixes and some increased capability was installed in July and a Technical Status Review and Requirements Review Board meeting (see Section 4) was held from July 25th to the 28th at WPAFB.
- As part of the ongoing test and evaluation process a revision of the build 1 software incorporating test user change requests was installed at the end of October.
- At the end of November a draft of a completely revised version of the Software Requirements Specification was submitted, and on November 30 and December 1 the second meeting of the Requirements Review Board was held at WPAFB.
- On December 16 build 2 of the software was delivered.
- During the first week in March 1995, build 3 of the software was installed and the Test Readiness Review was held at WPAFB. At that time the plan was to use the rest of Phase III to correct software errors and incorporate user change requests in preparation for Phase IV.
- In April BBN was notified that Phase IV would not be funded and that the focus of the rest of Phase III should be on integrating the OASYS system with the Operator Model Architecture (OMAR) model shell to support test and evaluation of human performance models developed in the OMAR environment. At the same time further work on the OASYS data-base was discontinued and several features were dropped from consideration.

As a result of this refocusing, the OASYS system was divided into two subsystems: the OASYS Task Analysis System (OASYS-TAS), which provides a capability for task and system-level analysis, and the OASYS Experiment System (OASYS-ES), which provides a capability for real-time human-in-the-loop and model-in-the-loop experimentation and data collection. OASYS-ES was "opened up" and reconceived as a set of tools that could be used together and in combination with COTS and custom software to facilitate operability experiments in a wider variety of technical circumstances. The final versions of the OASYS software (CDRL A013), including a

subsystem to support OMAR modeling and a subsystem for task timing and load analysis, were delivered to WPAFB in mid June along with User Manuals (CDRL A015), the Programmer Manual (CDRL A014), and Installation Procedures (CDRL A004). The OASYS Final Technical Review was held at WPAFB at that time.

#### **4. OASYS Prototype Test and Evaluation**

In order to support the prototype design approach for OASYS, the project needed a community of potential users who would review and try to use the design prototypes and provide continuous evaluation of the developing system. During Phase I and II of the program, the Air Force identified satellite control and specifically satellite control for the Brilliant Eyes program as a possible user community with potential operability problems.

In May of 1992 representatives from BBN and AL/HRG attended a meeting of the Human Computer Interface Working Group (HCIWG) at Pt. Mugu to investigate the possibility of using satellite control as a sample problem for evaluating OASYS. As a result of this visit, there was general agreement that satellite control would be an appropriate sample problem to use for OASYS evaluation and test.

In April 1993, Armstrong Laboratory and BBN gave a briefing and demonstrations to personnel from Air Force Space Command. Discussions at that time supported the belief that satellite control would be an appropriate test domain.

The Air Force next identified the Brilliant Eyes (BE) program as a specific satellite program that might work with the OASYS project. In June of 1993 a meeting was held at Kirtland Air Force Base where Air Force Space Command (AFSPACECOM) and Air Force Operational Test and Evaluation Center (AFOTEC) representatives were briefed on the OASYS project. Additional briefings and discussions were held with the BE program office in September.

As a result of these interactions, a Memorandum of Agreement was executed between AL/HRG, AFOTEC, AFSPACECOM, and the BE program office to support the development of OASYS and to apply the OASYS prototypes to BE operability. A Requirements Review Board (RRB) with representatives from each of these agencies was formed to provide user input to the project. The RRB met as part of the OASYS project reviews. At these meetings the RRB members were briefed on the latest design issues, saw demonstrations of prototype software, and participated in hands-on trials of the prototype software. After each meeting the RRB made recommendations for design changes or additions to improve the design.

In addition to the RRB reviews, human factors expert consultants to AL reviewed and tested each release and made recommendations for system improvements as well as identifying software errors.

A list of recommended changes due to either software errors or recommendations from the RRB or the AL human factors experts was maintained at both AL and at BBN. Weekly telephone conferences served the function of a change review board to monitor the status of entries on the change list. There were 231 proposed changes. Of these, approximately 180 either resulted in changes to the system or were made obsolete by major redesign of the related tool.

## **5. OASYS Capabilities—A User's Perspective**

OASYS (Operability Assessment System) is a suite of tools that supports the investigation of operability issues. We define operability assessment as the process of analyzing whether a *system* composed of automated capabilities and human operators interacting with those capabilities can successfully accomplish its intended mission.

This broad definition of operability assessment includes activities such as the analysis of tasks required to accomplish a mission, the analysis of tradeoffs in the allocation of these tasks to humans or machines, the analysis of the number of individuals needed to operate a system and their required skills and training, the design and development of prototypes of human-machine interfaces, the measurement of the workload and performance of human operators using these interfaces during simulated system operation, and the testing and validation of human-performance models of system operators.

Operability assessment can and should take place throughout the system design and development life cycle. As system requirements and system design are developed at successively deeper levels of detail, OASYS supports the development of increasingly detailed and realistic task specifications, operator workstation simulations, and operational concepts based on mission requirements.

This section describes OASYS from the user's point of view. The section begins with an overview of OASYS capabilities. It then provides examples of some real-world operability issues that might be resolved with the OASYS tools. Finally, it describes a methodology for operability assessment and describes the use of OASYS capabilities to implement that methodology.

## **5.1 Overview of OASYS Capabilities**

OASYS supplies two major types of capabilities—a capability for task and system-level analysis provided by the OASYS Task Analysis System (OASYS-TAS), and a capability for real-time human-in-the-loop and model-in-the-loop experimentation and data collection provided by the OASYS Experiment System (OASYS-ES). OASYS-TAS may be used to develop projections based on task decomposition in order to identify possible operability bottlenecks. Potential problem conditions can then be explored in depth by using OASYS-ES to simulate system operations and using test operators or human-operator models to generate data on human-system performance.

OASYS-TAS is a tool for finding potential operability problems based on task assignments, operator load, task coordination, and situation complexity. The user can develop an annotated hierarchical task structure, diagram this task structure in a flow chart, assign agents to perform each task, and designate task-completion times and operator loads for each task. Based on a user-developed script of the critical events that initiate tasks, OASYS-TAS will present a timeline view that shows where tasks overlap, where delays may occur, and where feasible operator loading levels have been exceeded.

OASYS-ES is a tool for performing operability demonstrations and experiments to collect human performance data. The user can develop models of system performance linked to human-interface prototypes; design, set up, and run experiments using those prototypes; and specify data to be collected on the performance of both human operators and human-operator models as they interact with the prototype interfaces. OASYS-ES supports team experiments with multiple operators. Also, human-in-the-loop and model-in-the-loop experiments may be run interchangeably or simultaneously—it is possible to run a multi-person team experiment with live test subjects in one or more operator positions and human-performance models in the other positions.

The OASYS tool suite is unique from several perspectives. First, it supplies an unusual breadth of capabilities. A multitude of tools exist for task decomposition, work flow modeling, building GUI prototypes, and building system simulations. OASYS provides all of these capabilities in one tool set geared toward operability assessment. It is specifically designed to support the systematic collection of operator performance data. OASYS is also unique in its support of human performance model test and evaluation. The OASYS simulation environment is designed for full substitutability of operator models and human operators—prototype interfaces may be operated by either humans or models, with identical performance data collected for both. This capability supports

testing and validation of human performance models through comparison with actual human performance on identical tasks. The use of human performance models as surrogates for operator testing offers the possibility of inexpensive operability evaluation early in the design process.

## **5.2 Using OASYS to Resolve Operability Issues**

This subsection describes how OASYS may be used for operability assessment. The method for using OASYS is described as a sequence of steps, beginning with task analysis, and culminating in the construction and use of human-in-the-loop and model-in-the-loop simulations for operability testing. While the steps as described are cumulative, each one can be performed independently in order to focus on a specific operability issue if sufficient input is available. Subsequent sections provide more detail on the OASYS capabilities relevant at each step.

This section is intended as a description of how to use the OASYS tools to assess operability, not as an extensive tutorial in operability assessment methods. A number of excellent references and handbooks exist on various aspects of operability analysis, including:

- Helander, M., Editor. (1988). *Handbook of Human-Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands.
- Lysaght, R.J., Hill, S.G., Dick, A.O., Plamondon, B.D., Linton, P.M., Wierwille, W.W., Zakland, A.L., Bittner, A.C., Jr. and Wherry, R.J. (1989). *Operator Workload: Comprehensive Review and Evaluation of Operator Workload Methodologies*, Technical Report 851, United States Army Research Institute for the Behavioral and Social Sciences, Alexandria, VA.
- Meister, D. (1985). *Behavioral Analysis and Measurement Methods*. John Wiley & Sons, New York, NY.
- Preece, J. (1994). *Human-Computer Interaction*. Reading, MA: Addison Wesley
- Shneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Reading, MA: Addison Wesley.
- Smith, S.L. and Mosier, J.N. (1986). *Guidelines for Designing User Interface Software*, MITRE Corporation, Bedford, MA.

### **5.2.1 The Challenge of Operability Assessment**

Military Standard MIL-H-46855B, Human Engineering Requirements for Military Systems, Equipment, and Facilities describes the analysis, design, and test process to be performed in order to insure that military systems, equipment, and facilities achieve effective integration of personnel into the design of the system. The process starts with a

mission analysis and a baseline scenario in order to identify the functions that must be performed to achieve mission objectives. Functions are to be allocated to human operators or to automated processes, and further decomposed into the specific tasks that must be performed to accomplish the functions. Information flows between tasks are analyzed, and each task is analyzed to determine its human-performance parameters, and, if possible, these parameters are quantified in a way that permits the effectiveness of human-computer interfaces to be studied in relation to total system performance. The analysis should identify any high risk areas for operability.

This task analysis is converted into detailed design and development plans to create human-computer interfaces that meet mission requirements. Studies and experiments are to be performed to resolve specific problems or issues. At the earliest possible point, mockups and models are to be constructed and dynamic simulation techniques used to resolve design issues. Eventually, test and evaluation is conducted to verify that the design meets human engineering criteria and overall system requirements.

In practice, the implementation of an operability assessment process such as that described in MIL-H-46855B presents a number of challenges to the operability analyst. One of the major obstacles to task analysis, especially early in system design, is the lack of detailed information about operator tasks. Requirements documents and high-level design documents are often vague regarding the specifics of human-operator responsibilities. The information that is available about operator tasks is often scattered across a number of diverse sources, each of which is incomplete by itself.

Furthermore, much critical information is not in written form, but exists only in the heads of a multitude of individuals. For example, one useful source of information about operator tasks for a new or upgraded system is the written documentation and training materials for other, similar systems, as well as the personal experiences of individuals who have operated these systems. Also, individuals working on the system design team often have a concept of how the system will operate that underlies their design work, and this implicit concept has implications for the tasks of the human operator. These operational concepts may not be precisely defined, however, and they may be limited to the part of the system on which the designer is focusing. Each individual may have his or her own perspective on how the system will function from the human operator's point of view. Operational concepts that are difficult to specify are even more difficult to quantify. Efforts to assess system operability constantly seek to define operator tasks precisely enough to allow quantitative measurement of task performance.

Even when relatively complete written descriptions of system functions exist, it can be very difficult to visualize the operator's perspective from written descriptions without actual experience in performing the tasks. Also, quantitative time requirements for tasks are very difficult to specify without experience. As one test operator, who was trying to complete a difficult task within the required time, commented: "I had no idea that 45 seconds was so short!" Prototypes that capture system functionality from the point of view of the operator can be very useful in supporting operability analysis, including the quantitative measurement of task time requirements. Such prototypes are frequently time consuming and costly to build, however, especially when systems functions are complex and for new systems where system functionality has not yet been well defined.

#### The Importance of Communication

Because there is often no single complete source of information that can be used to build a description of operator tasks, because much useful information may not be in written form, and because individuals are likely to have many different perspectives on system operation, effective communication is essential to successful operability assessment.

Typically, the operability analyst cannot develop a task analysis without the input of many different individuals. Task descriptions and task flows should be developed through an iterative review and verification cycle that involves the whole system-design team, not just the operability analyst, as well as individuals who have operational experience with other similar systems.

A documentation and description mechanism is needed to facilitate this communication and review process. Hierarchical trees can be used to show how a series of tasks relates to a higher-level function, and flow charting techniques can be useful for capturing sequential and parallel relationships among tasks.

Prototypes of human-computer interfaces are also a useful tool for fostering communication as the system design is developed. Concrete representations of system functionality from the operator's viewpoint can be used as a starting point for discussions of the adequacy of system operability concepts. Early prototypes can reveal major operability problems. As the design progresses, higher fidelity prototypes can be used to collect quantitative performance data to ensure that operator performance is adequate and that workload is within feasible levels.

The OASYS tools have been designed to support communication throughout the operability assessment process. Diagrams showing the decomposition of functions into



tasks and the sequence of those tasks may be developed using OASYS-TAS. Changes may be made as the descriptions are reviewed and revised, producing a task description that captures information from many different sources. OASYS-ES is a tool for building dynamic interface prototypes, driven by scripts and system simulations. These dynamic prototypes may be used as a basis for discussion in walkthroughs and demonstrations or may be used to collect both qualitative and quantitative data in systematic controlled experiments.

### Repositories of Operability Information

Operability information for a new or upgraded systems comes both from work with existing systems that are performing similar missions, and from the design, development, and testing of the new system. Operability information from both sources is typically stored in four types of repositories:

- **Written and Graphical System Descriptions.** This includes documents such as Mission Needs Statements, requirements specifications, Concept of Operations (CONOPS) documents, design documents, and training materials.
- **Numerical Databases.** This includes historical performance, workload, and staffing data for similar systems as well as experiment and test results for the system being designed.
- **Existing Systems or Working Prototypes of New Systems.** The system itself is an important repository of information. Existing systems that perform similar missions are a rich source of operability information for new systems, although the new system may differ from the older one in significant ways. Prototypes of the new system typically capture only a portion of the functionality of the full system, but they embody information in a dynamic way that is qualitatively different from other information repositories.
- **Expert Knowledge.** The minds of system builders, analysts, and potential users are an important repository of knowledge about how the system may be operated and the operability issues that may prove to be important.

The use of OASYS for operability assessment has two goals: 1) to increase the *amount* of information in each of these repositories, and 2) to increase the *accessibility* of this information.

Use of OASYS can increase the amount of operability information available in each type of repository. OASYS-TAS can be used to develop task decompositions and task sequences, increasing the amount of descriptive operability information available. Use of OASYS-ES supports the translation of written documents into working prototypes, and these prototypes provide a means for collecting numerical performance and workload data. The process of developing task descriptions and task sequences, building

prototypes, and using those prototypes to collect data increases the amount of expert knowledge available. The building and use of prototypes also promotes shared operability awareness and congruent understanding of operational concepts among system builders and users.

The use of OASYS can also increase information accessibility. The development of task decompositions and task sequences based on multi-person input consolidates information from multiple individuals in one accessible location. The translation of written requirements and designs into working prototypes makes the design more accessible and more understandable to potential operators of the system. Finally, lessons learned from building and testing prototype interfaces can be incorporated back into system design documents for broad distribution to the design team. Ultimately, a system specification can include both written documents and a working prototype, with each one capturing different aspects of the operational system.

### 5.2.2 Overview of OASYS Use

Figure 1 provides an overview of the use of OASYS tools throughout an operability assessment process. The process begins with the decomposition of functions into tasks, which are specified at a detailed level. The flow or sequence tasks of these tasks is then specified and an allocation of tasks to human operators is developed.

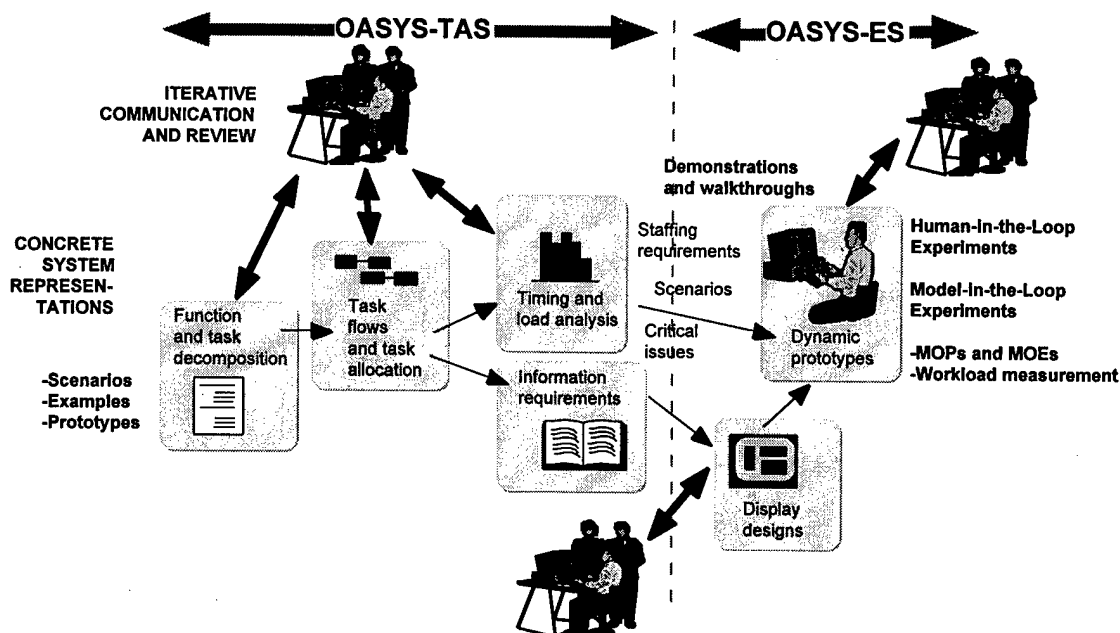


Figure 1. Using OASYS for Operability Assessment

If the amount of operator time needed to perform each task can be estimated, then a time and load analysis can be performed to determine if a defined task sequence can be performed without overloading the operators under a specified task allocation. If task time requirements cannot be estimated based on existing systems, it may be necessary to collect data on the time needed by operators to perform tasks using dynamic prototypes of the system. Based on task timing data, operator time requirements may be analyzed under a variety of scenarios to determine if staffing levels will be adequate for different types of situations that place different demands on the operators. This time and load analysis may identify critical issues that require more in-depth analysis and data collection using a dynamic prototype of the planned interface.

Based on the task specification, information requirements can be developed for operator-machine interfaces, i.e., what information will be needed to perform the tasks? These information requirements then form the basis for the design of prototype displays. Finally, a simulation is built to make the displays dynamic, using scenarios to specify the timing and frequency of events external to the system.

This dynamic prototype may be used for demonstrations to elicit comments from potential system users. Prototypes may also be used in controlled human-in-the-loop experiments to collect performance and workload data. Task-time data from experiments may be used to update time estimates in the time and load analysis, resulting in more accurate timelines for task completion and more accurate estimates of the number of individuals needed to operate the system. Human-in-the-loop experiments may also be used to assess skill requirements for operator positions by measuring the performance of individuals with different skill levels as they interact with the dynamic prototype. The use of OASYS at each of these steps in operability assessment is discussed in more detail below.

### **5.2.3 Developing Function and Task Decompositions**

The purpose of function and task decomposition is to identify and describe the potential roles and tasks to be performed by human operators. The information available for task decomposition will depend on the nature of the operability assessment and the stage of system development. For example, a new system that is still in the concept development stage may have relatively few sources of information about the tasks of the human operator, while a planned upgrade to an existing system may have extensive information on operator as well as system functions and tasks. Possible sources of information for building a function and task decomposition include:

- Mission Needs Statements. These will typically describe high-level system functions, but may include scenario descriptions that are useful in developing task descriptions.
- Concept of Operations (CONOPS) documents. These will typically describe the functions of the human operator, although they may not be at a detailed level.
- Requirements documents. These will often describe requirements for overall system performance, but may not provide detail on human operator performance requirements.
- Design documents. Depending on the stage of the design process, these documents may provide detailed descriptions of system design. Information about operator tasks is often implicit in these descriptions, but further work may be needed to develop explicit descriptions of operator tasks.
- The design and development team. The individuals (typically engineers) working on the design of a new or upgraded system will often have implicit concepts about the operation of the system, and, as a consequence, the tasks of the human operator. These implicit concepts may be made explicit through iterative review of the task descriptions and task sequences developed by the operability analyst. Even though the design and development engineers may not be able to generate operator task descriptions, they often can critique such descriptions effectively ("the operator won't be able to do that because the system....").
- Documentation for similar systems, especially CONOPS and training materials. Most new systems are similar enough to existing systems that there is a substantial overlap in the responsibilities and tasks of the human operators. Existing systems that have been in operation for some years typically have operational concepts and detailed training materials that are a rich source of information on operator tasks.
- Individuals who have experience with similar existing systems. Individuals who have operated similar systems are an excellent source of information on operator tasks as well as operability issues. These tasks may need to be modified for a new system, however.

To what level of detail should task decompositions be taken? In practice, the useful level of detail depends on the objectives. Singleton (1994) comments that "How far to take the task analysis is a function of the accessible data and the time available." As a rule of thumb, task analysis needs to be taken to the level at which the following questions can be answered:

- What decisions and actions are associated with the task?
- What events trigger the task, i.e., when must the task be performed?
- What are the closing events for the tasks, i.e., when is the task completed?
- What are the inputs to the task, e.g., what information is required?

- What are the outputs or results of the task?
- What are the subsequent tasks, i.e., what happens next?
- How much operator time is needed to complete the task?
- Is there a possibility of automating the task?
- Who might perform this task? What kind of skills or experience might be required?

For example, in a decomposition of the operator tasks performed in a satellite control center, "perform satellite support" may be such a high-level task that it makes no sense to try to answer the above questions, whereas "contact satellite ground station" (a subtask of perform satellite support) is a specific-enough task that the questions may be answered

#### Using OASYS to Develop Function and Task Decompositions

OASYS-TAS provides a graphical tool for drawing hierarchical function and task decompositions. The basic building block of task decomposition in OASYS-TAS is a box, which represents an "activity." For each activity, the analyst specifies a name for the activity, and, if desired, an identifier for the type of individual (or automated process) performing the activity. Each activity box may contain multiple subboxes to represent subtasks. There is no preset limit to the number of levels to which a decomposition may be carried in OASYS. The analyst may easily move up and down the task hierarchy, adding or removing tasks as required. Only one level of the decomposition may be seen on the screen at a given time, however. The analyst may print each level of the decomposition to produce hard copy for review by others.

For each task (at any level in the hierarchy), the analyst may enter a series of task attributes that describe the task, corresponding roughly to the questions listed above. These attributes serve to provide more detail about the task. Time estimates for tasks, if provided, may be used as inputs for the OASYS-TAS time and load analysis capability described below.

Typically, an operability analyst might start to develop a task decomposition early in the design and development process, perhaps in the concept development phase. The analyst would start with the printed documentation about the system, including the Mission Needs Statement, any requirements documents, the CONOPS, if available, and any other information available. If available, training and operations manuals from similar systems will be very helpful.

The analyst will then develop a tentative task hierarchy and create a graphic representation of this hierarchy using OASYS-TAS. This hierarchy is likely to have a

number of areas in which the necessary level of detail is missing. The analyst will then seek further information from other members of the design and development team, and from individuals who have experience with similar systems. These individuals will be asked to review, correct, and add detail to the task decomposition. This may be done directly by bringing up the OASYS-TAS display of the task hierarchy and making changes as they are suggested, or it may be done remotely using paper copies of the task decomposition and making changes to the OASYS-TAS decomposition afterwards.

Task decomposition is most easily considered in conjunction with task flow (see below). It is typically easier to develop a list of the tasks required to operate a system by using concrete examples—scenarios—than by attempting to list all tasks independent of their interrelationships and without considering the external events that drive the tasks. One or more specific scenarios can be used to set the stage for the task decomposition by specifying a series of external events that drive the tasks, e.g., when event A occurs, the operator will do X, then Y, and then Z. A number of different scenarios may be needed to ensure that the task decomposition includes all of the tasks required under different circumstances, and multiple individuals should be consulted in developing the task decomposition. As each individual's input is incorporated, the task decomposition will become more detailed. Ultimately, the decomposition should reflect a consensus about the tasks required to operate the new or upgraded system.

#### **5.2.4 Specifying Task Flows and Task Allocations**

Operator tasks are rarely independent of one another—instead they exist in a complex network of logical interdependency, with the start of one task often dependent on the completion of another. Sequential interdependencies are often identified as part of the task decomposition process, e.g., “the next logical step after contacting the ground station is to send antenna-pointing instructions to the ground station.” Tasks that are parallel, rather than sequential, can often be identified by working backwards from a mutual point of completion, e.g., “in order for this task to be initiated, these two tasks must be completed, but the two prior tasks do not depend on one another.” Tasks may also be conditionally dependent, e.g., the operator initiates a repair task only if a fault is detected during a maintenance check. Sequential and conditional dependencies among tasks are captured in a task flow, which documents the sequence in which tasks are performed.

External events may also drive the sequence in which tasks are performed. Some tasks may be on-going and independent of external events, e.g., a system status check performed once an hour. Other tasks may not be performed at all unless an external event occurs, e.g., the processing of an incoming message. Any specification of task flows

needs an underlying scenario or script of external events to drive it by specifying the occurrence of those events that initiate task sequences.

#### Using OASYS-TAS to Document Task Flows

OASYS-TAS provides a graphic flow-charting capability for documenting task interdependencies. The flow chart is created by selecting icons that represent "nodes"—activities, decisions, or events—from a palette and linking them together to create sequential flows and dependencies among tasks. In addition to the activity nodes described above, the analyst may create decision nodes that specify possible branching to other tasks as the result of a decision, and AND and OR nodes (from standard flow-chart terminology) that incorporate logical rules for task flows. Events that initiate a task or a series of tasks are indicated by an Init Event element in the flowchart that shows the point in the task flow at which the initiating event occurs. All of the nodes and events in the task flow may be connected by arrows, showing the sequence in which the tasks are executed.

Typically, the analyst will document the task flow as the task decomposition is developed. That is, as tasks are identified, their relationship to other tasks will be specified. While OASYS-TAS allows the creation of unconnected nodes and events, the analyst will rarely create such isolated nodes. The logical flow between activities will be an integral part of developing a task decomposition based on a scenario, e.g., "as soon as the B event occurs, the operator will begin doing M. As soon as that task is completed, the operator will proceed to do N. At the same time, task P must be performed on an ongoing basis."

The goal of the task flow chart is to be as specific as possible and to communicate clearly. Logical flows between tasks as documented in the task flow will be reviewed as part of the task decomposition by other members of the design and development team as well as by experts in similar systems. Once tasks are allocated to operators (or to automated functions) the task flow chart also forms the basis for a time and load analysis.

#### Allocation of Tasks to Humans or Machines

The next step in the task analysis is to allocate the tasks to human operators or to automated processing. While many system functions and tasks are clearly automated, or clearly the responsibility of the human operator, some tasks may fall into an intermediate "gray" area where human/machine responsibilities are unclear. These tasks require further analysis to determine the best mix of human and machine capabilities.

Classic human factors guidelines call for tasks to be allocated to humans or machines based on the relative strengths and capabilities of each. A recent review of actual practice, however, argues that allocation cannot be separated from design, and that the system-performance effects of hypothetical allocations of functions to humans or machines are often impossible to model (Fuld, 1993).

Increased levels of automation are often an intrinsic part of the concept for a new or upgraded system. A realistic operational allocation question is usually of the form: "we are planning to automate the BB process, but are unsure about how much and what kind of information to give the operator, how much control the operator should have, and when the operator should intervene in the automated process." Thus the operability issue for human/machine allocation is not "yes or no," but the degree and type of control over the automated process that should be provided to the operator.

### Allocation of Tasks to Human Operators

In practice, a tentative allocation of tasks to human operators is usually developed, tested, and refined as a system is designed and built. Initial allocations are often based on:

- Current practice in existing systems. If similar positions exist in an operational system, the responsibilities (and skill levels) associated with those positions are a good starting point for the definition of positions in the new system.
- CONOPS. If a CONOPS document exists, it may specify operator positions.
- Physical space constraints. Under some circumstances, the number of operators is constrained by the space available, e.g., a mobile command center may only have room for two or three operators. In this case, the issue is not how many operators are needed, but whether the available operators can handle the task load.
- Available personnel. The allocation of tasks to operators may be constrained by the knowledge that only certain types or numbers of individuals will be available to operate the system.
- Lines of command and authority. Because of their consequences, certain responsibilities and decisions may be restricted to a particular military rank or authority level.

### Allocation of Functions and Tasks in OASYS-TAS

Human interactions with automated processes may be analyzed in OASYS through the development of detailed task decompositions and task flows for automation concepts. Note that the task decompositions for different degrees of automation are likely to have different structures, not just different allocations of tasks to humans or machines. That is, the introduction of automation usually means that functions and tasks are divided and



organized differently, not simply that the machine performs tasks formerly performed by the human. Based on these task flows, the information needed by the human operator interacting with the automated system can be identified, and dynamic prototypes developed to provide that information. Automation concepts can then be tested by measuring the performance of operators as they interact with (simulated) automated systems in realistic scenarios. Note that the simulation need not actually include the automated systems being contemplated; it simply must reproduce the behavior of the automated processes from the operator's point of view in a specific scenario. The results of this testing can then be used to improve the concept for human-automation interaction.

In assigning tasks to operators, the first step is to make a tentative assignment of tasks in a task flow diagram to a type of operator (e.g., "satellite controller") and specify these assignments as an attribute of each task in the OASYS-TAS task flow. If task-completion times can be estimated, a timing and load analysis (see below) can be conducted using a variety of scenario scripts to determine if, and when, system operators will be overloaded and which operators will have an unmanageable volume of work. If operator load appears to be a problem, task-allocation concepts can be further tested with rapid prototypes to ensure that a single operator can handle the tasks allocated to him or her under a variety of conditions. For a team of operators, an OASYS-ES prototype may be used to test whether a team of operators can meet mission requirements under a candidate allocation of tasks.

Skill-level requirements for operator positions may also be an operability issue for new or upgraded systems. Assessment of the skill level required for a position is typically based on knowledge about existing similar systems, e.g., similar tasks are performed in the AAA system by individuals with BBB specialized training or experience. In the rare instances where a new system requires operator tasks that have little or no resemblance to tasks in existing systems, it may be necessary to use dynamic prototypes to test the ability of operators with different skill levels, experience, or training to perform the task.

### **5.2.5 Analyzing Task Timing and Operator Loads**

Task timing and task allocation fit together dynamically to determine system operability. The adequacy of available time depends on what tasks must be accomplished in that time period and how many people are available to accomplish them. In the same way, the adequacy of the crew size depends on the number of tasks to be accomplished and the time available to accomplish them. Operability analysis often fixes one of these

factors and then analyzes the other, asking how much time is required or how many operators are needed.

Answers to questions such as "Can the tasks required to contact, check status, and download information from a satellite be accomplished in a 20-minute time window?" or "Can a crew of two operators in a command vehicle operate a surface-to-air missile system to intercept an incoming TBM?" require the specification of the following information:

- What are the sequential dependencies of the tasks? Must some tasks be completed before others can be started?
- How much time is required to accomplish each task? This may be subject to external constraints (e.g., the task cannot be finished until an acknowledgment is received) or it may depend completely on how quickly the operator can act.
- Does each task occupy one hundred percent of the operator's time and attention, or is it possible that some tasks may be accomplished simultaneously?
- How are tasks allocated to operators?
- What (if any) external events trigger the tasks? Under what scenario of external events is the analysis being conducted?

If each of these factors can be specified, it is possible to create a timeline of tasks to be accomplished by each operator under a given scenario, and to calculate the total "load" of the operator, e.g., is the operator being scheduled to accomplish two full-time tasks simultaneously?

#### Using OASYS-TAS to Analyze Task Timing and Operator Load

OASYS-TAS task flow diagrams allow the analyst to specify task dependencies (i.e., one task cannot start until another is completed), the time required to complete each task, and the type of operator to whom the task is to be assigned. Based on this information, and on a script that specifies external events, OASYS-TAS calculates and presents a graphic timeline view of the tasks along with a graph indicating the load experienced by each operator.

Task times may be specified using two parameters: 1) the task execution time, and 2) the percentage of the operator's capabilities that are required during that time. For example, a task that occupied the operator completely (i.e., no other tasks could be done) for five minutes would be designated as having a five-minute execution time at 100 percent load. A task that occupied the operator intermittently over a one-hour period (e.g., a status monitoring task) might be designated as having a 60-minute execution time but only a 50 percent load.

Scripts contain external events that may initiate the start of a task or a sequence of tasks. The scripts specify the time that the event occurs, as well as the operator who will handle that event. Scripts may also be used to inject events that cause the task flow to follow a particular path. For example, if "check for faults" is a task, then there may be two subsequent task flows: one if a fault is found and one if no fault is found. Scripts may be used to trigger one of these two paths.

A typical use of the timing and load analysis capabilities might be to examine the load for an operator performing a number of different tasks over a period of one to two hours, during which a number of external events occur. The analysis might show that the operator is fully occupied for most of that period, has some "down time" periods, and has some periods in which the analysis shows a load of greater than 100 percent, i.e., more tasks than it is possible to accomplish.

The analyst would then examine the tasks being performed during the overload period and consider whether some of these tasks could be moved to slower periods. If there is no possibility of moving the tasks, then another operator may be required. A new operator could be added, tasks allocated to the new operator, and the analysis rerun to see if load for the two operators remains within feasible bounds throughout the period. The two-operator task allocation could then be tested using a more demanding script (in which there are more external events triggering tasks) to see if two operators will be adequate for extreme conditions.

Timing and load analysis may be used to identify tasks and scenarios that place high demands on system operators, i.e., to identify those situations in which the operability of the system is most in question. Prototyping and experiment efforts can then focus on those areas that are most difficult for the operator.

The most challenging aspect of timing and load analysis is the specification of task times. It can be very difficult to estimate task times early in a design process for a new system. For tasks that resemble those already performed in existing systems, task-time databases may be available. Also, operators of existing systems may be able to provide estimates ("How long would it take you to....."). For new tasks with little resemblance to existing tasks, it may be necessary to make assumptions about the time needed to perform the task, and then test those assumptions by collecting task-time data using dynamic prototypes in human-in-the-loop experiments.

### **5.2.6 Specifying Information Requirements**

Information requirements are the bridge between task decompositions, task flows, and display designs. Most tasks will have a set of information that is required to perform that task. If the task involves making a decision, then information will often be required to make that decision. Much of the information needed to perform tasks and make decisions comes from the human-computer interface. Therefore, the interface must be designed so that the operator has access to the required information *at the time that it is needed*.

Computer displays are of finite size, and can only contain a limited amount of information at any one time. Therefore, an interface design strategy is needed to ensure that the display changes dynamically as the task and associated information requirements change. This may be done by automatic changes, e.g., certain information always appears when a certain type of message is received, or it may be under the control of the operator. In either case, the interface designer must be careful that the operator does not have to constantly change displays in a search for the relevant information for a particular task.

Information requirements for tasks are typically identified through a combination of analysis of existing systems, interviews with experienced operators, and common sense. It is critical to have a detailed task decomposition to support this process because information requirements can be identified sensibly only at a very detailed level. Descriptions of specific situations are also helpful in identifying information requirements, i.e., "what information is needed to perform this task in this situation?"

#### Specifying Information Requirements Using OASYS-TAS

Information requirements are very difficult to specify at a high level of task generality (e.g., "perform satellite support"). Detailed task decompositions developed using OASYS-TAS support the process of specifying the information required to perform those tasks. As information requirements are developed, OASYS-TAS provides a way to record and document those requirements by allowing the analyst to specify "input information" as an attribute of each task.

Task timelines developed using the OASYS-TAS timing and load analysis capabilities may also be helpful in grouping information requirements for tasks that typically occur in close proximity to one another in time. These grouped information requirements then form the basis for design of displays to support those tasks, preventing operators from having to switch rapidly among many different displays.

### **5.2.7 Selecting an Area for Prototyping**

Early in the design and development process, it may not be feasible to develop a prototype of an entire system. Instead, portions of system functionality are typically selected for prototype development. Often, the areas selected for prototyping are those considered to be "high risk," i.e., areas in which there are serious questions or issues to be resolved before design or development can proceed.

Where are the high risk areas for operability? How can they be identified early in the design process? One way to identify potential operability issues is to concentrate on those aspects of the systems that are most different from existing systems. New features, functions, and tasks are most likely to generate new operability issues. Of course, if similar existing systems have operability problems, then these problems should be considering in designing the new system.

OASYS-TAS timing and load analysis may also identify possible "problem spots" where operators may be overloaded and operability problems may arise. Circumstances in which a number of tasks must be performed simultaneously or where complex tasks must be completed in very brief time periods are natural candidates for further analysis and testing through prototype development.

### **5.2.8 Developing Display Designs**

Display designs are based on analysis of the tasks to be performed and the information required for those tasks. Information requirements are then grouped by operator to create displays appropriate to operator positions in the system, and, to the extent possible, grouped by task sequence so that the information needed for related tasks will be easily accessible.

After the information needed for each task has been identified, the next step is to develop display methods for each type of information. Ideas for display methods come from a variety of sources, including the displays that have been successfully used in other systems, requirements for consistency with other systems, and relevant human factors standards and guidelines. Creative ideas for information display may be developed, but care must be taken to ensure that innovative displays are truly useful to the operator and not just a showcase for new technology. Prototype development and testing of innovative displays with individuals who are representative of system operators should be conducted to evaluate the usability of the displays.

It is usually helpful to sketch prototype designs on paper and then use a graphical user interface (GUI) building tool to compare alternative layouts within realistic screen real estate constraints. Static prototypes in the form of storyboards can be used to gather initial opinions about the GUI design from system operators.

An important element in choosing the displays to be built is the purpose of the demonstration or experiment for which the displays are to be used. The operability analyst should be able to clearly describe the purpose of the demonstration or the hypotheses to be tested in the experiment. The lessons to be learned from the prototype designs will determine which displays need to be developed and what information and capabilities need to be included on the display prototypes. For example, if the purpose of the prototype is to test alternative graphical methods for presenting information about a satellite's health and status, there may be no need to develop the displays that support ground-station contact procedures.

#### Building a GUI with OASYS-ES

OASYS-ES provides a GUI-building capability that allows the analyst to quickly create a facade that shows the static appearance of the display. A library of gadgets is available for commonly used elements of screen design such as buttons, pulldown menus, and windows that contain text. A screen design is created by selecting these gadgets from an icon palette and dragging them into the desired position on the screen. Gadgets may be edited to change their position, color, size, or font type as needed. New gadgets may be created and added to the library to create display objects that are specific to a particular application.

The user assigns a name to each gadget on the display. This name is then used in creating a simulation to drive the display, allowing the operator to provide input to or receive output from the simulation through the GUI.

### **5.2.9 Building and Using Dynamic Prototypes in Demonstrations and Experiments**

Dynamic prototypes of a system are invaluable in operability assessment. Prototypes that show how the system will behave in a specific situation and allow the operator to interact with the system in realistic circumstances can be used to detect operability problems and to resolve issues before full-scale development begins.

The key to making dynamic prototypes valuable for operability assessment is to have clearly defined goals for the *use* of those prototypes. It is not sufficient simply to build a prototype—the purposes of that prototype in terms of operability assessment must be specified from the beginning. The operator tasks that are supported by the prototype

must be clearly specified, or there will be no way to assess whether the prototype is successful from an operability standpoint. For example, if a GUI is designed to support certain operator tasks, then a dynamic interactive prototype of that GUI can be used to assess whether operators are able to successfully perform those tasks using the GUI. If the purpose of the prototype GUI is not well defined, however, then there are no obvious criteria for judging the success of the prototype from an operability standpoint.

The level of fidelity of simulated system behavior should be driven by the purpose of the simulation and the desired behavior of the GUI. Simulation design issues may include the appropriate granularity of time units (e.g., seconds or minutes), the importance of including realistic detail, and the need to introduce errors or uncertainties. For operability assessment, there is no reason to build into the simulation detail and fidelity that will not be apparent to the human operator.

Dynamic prototypes may be used to gather operability information through demonstrations or through human-in-the-loop experiments. Demonstrations are typically less formal than experiments. They are designed to elicit feedback from operators, but not to collect objective quantitative data on operator performance measures such as speed or accuracy. Feedback may be in the form of verbal comments, preferences, or ratings of the value of different GUI features. If operators do not have a chance to actually interact with the system in trying to accomplish specific tasks, however, their feedback may not be very detailed or specific.

Human-in-the-loop experiments are typically designed to collect quantitative data on operator performance and operator workload, as well feedback on operator preferences and the value of display features. In order to have confidence in the quantitative performance and workload data collected, it is usually necessary to use a number of different operators as well as multiple scenarios in the experiment.

#### Using OASYS-ES to Build a System Model and a Script

A major feature of OASYS-ES is the capability to build simulations of system behavior in order to produce dynamic interactive GUIs for operability assessment. The basic building block of the system model is the Event Translator, which reacts to input events in order to produce output events.

Working from the GUI, the modeler identifies the desired consequences of each operator interaction with an input gadget, and the information to be displayed in each output gadget. The modeler then defines a series of Event Translators that produce the

desired behavior, i.e., react to the GUI input gadgets and provide information to the GUI output gadgets.

One or more scripts are developed that specify the external (outside the system) events that drive the system model. For example, the approach of a hostile airplane might be part of script for a surface-to-air missile system. The detection of that plane, the sending of an alert message to the operator, and the response to the operator's commands to intercept the plane would be simulated with Event Translators in the system model. This use of scripts allows testing of the system in many different types of situations without the need to change the system model.

#### Using OASYS-ES for Demonstrations

The primary purpose of demonstrations is communication and feedback. Demonstrations can serve to educate system designers about operability issues, and to elicit opinions about GUI designs from system operators. Demonstrations are typically brief and involve only one or a few scenarios. If the audience for the demonstration is playing only a passive role, i.e., is not interacting with the prototype, then it may be difficult to hold their attention for a longer period. Data collection during demonstrations is primarily qualitative and anecdotal, and is most useful for early detection of major problems. Feedback may be used to modify display designs before more extensive testing is conducted.

OASYS-ES may be used to conduct demonstrations using a subset of the capability provided for conducting experiments (discussed below). A demonstration is implemented as a single experiment trial. The user specifies a GUI, system model, and script for the trial and runs the trial to begin the demonstration. The user would probably choose not to collect performance data during the trial.

#### Using OASYS-ES for Human-in-the-Loop Experiments

The primary purpose of human-in-the-loop experiments is to resolve operability issues that require collection of data on well-defined measures of performance or effectiveness. Typical questions include: Can operators perform this task quickly enough to meet time limits? Can operators achieve the needed degree of accuracy in their decisions? or Can operators maintain a feasible level of workload in stressful situations?

Performance measures often collected in operability experiments include the time needed to complete tasks, the frequency of correct decisions, and the output of a task (number of actions completed during a time period). Data for these measures may be collected using OASYS-ES. Data for other measures may be collected through paper-



and-pencil questionnaires, including subjective workload ratings if operator load is an operability concern for the system, or usability measures such as user evaluations of the adequacy and value of display features. Data for interface-usage measures such as the number of times the operator switched between displays may also be collected using OASYS-ES.

Experiments are typically designed to focus on a limited number of hypotheses, to control sources of variability other than those of interest, and to provide sufficient data for statistical tests of hypotheses. Experiments are often designed to compare the effects of two or more factors that are systematically varied in the experiment, i.e., independent variables.

OASYS-ES operability experiments may focus one or more types of independent variables. The major types likely to be interest for operability are:

- Variables associated with the design of the interface. These will be implemented in OASYS-ES as different GUIs. For example, the user may want to run an experiment to compare performance with two or more different GUI designs.
- Variables associated with the capabilities of the system. These will be implemented in OASYS-ES as different system models. For example, the user may want to run an experiment comparing the operability of a system under two or more different assumptions about its performance, e.g., when information will be available, how accurate it will be, etc. As another example, the user may want to test different levels or types of automated capability, also requiring two different system models.
- Variables associated with the script. For example, the user may want to assess operability under low-stress and high-stress situations caused by external events.
- Variables associated with individual subjects or teams of subjects. For example, the user may want to compare the performance of individuals with and without specialized training, or the performance of experienced and inexperienced teams.

Trials in an experiment are defined by specifying a value for each of the independent variables listed above: the GUI to be used, the system model, the script, and the subject or team of subjects. In order to design and run an experiment using OASYS-ES, the analyst sets up an experiment design table. This experiment design table specifies the value for each independent variable in each trial, the number of trials to be run, and the data to be collected.

The experiment design contains a data collection plan that specifies which OASYS simulation events are to be collected in a data file. Each event is time-tagged. Data files may be created for each trial in the experiment, or one file may be created for the entire

experiment. After data files have been completed, they may be exported for analysis with the Statistical Analysis Systems (SAS).

Note that in order for the appropriate data to be collected during a simulation run, the modeler who creates the OASYS-ES simulation must build into the simulation appropriate events that can be used as the basis for data collection. For example, if the analyst wishes to measure the time that elapses between the receipt of a message by the operator and an operator action (e.g., clicking on a button) based on that message, the modeler must make sure that the receipt of the message and the button click are associated with simulation events that can be collected in a data file.

Experiment data have multiple uses for operability assessment. They may be used to resolve specific operability issues, e.g., can a task be performed adequately by a team of two operators within a specified time period? Experiments are also a source of task-completion time data, which may be used in timing and load analyses, added to historical task-time databases, and used as input to staffing and training models. Experiment results may also be relevant for defining training and manpower requirements by providing evidence about the number of operators needed for a system, the adequacy of performance of operators at different skill or proficiency levels, and the amount of training needed to perform certain tasks.

#### Using OASYS-ES for Model-in-the-Loop Experiments

The development and use of human performance models offers great potential for reducing the cost and time associated with testing human subjects during operability assessment. To the extent that human performance can be accurately modeled, early operability testing could be performed through model-in-the-loop simulation. A combination of human-in-the-loop and model-in-the-loop testing is also useful for operability assessment in multi-person tasks, reducing the number of human operators needed for meaningful testing.

Human performance models to be used for operability assessment must be thoroughly tested and validated through the comparison of model performance data with human performance data. OASYS-ES has unique capabilities for the testing and validation of such models. Models may be substituted for human operators at any operator position in an OASYS-ES simulation, and identical performance data collected for models and human subjects performing identical tasks. This data-collection capability supports a detailed comparison of the model's performance to that of human operators. OASYS-ES also supports experiments involving teams of operators using multiple workstations, with models substituted for human operators at any or all of the operator positions. This

allows testing of individual operator performance in a multi-person team task without the need to involve multiple individuals in each test session.

## **6. OASYS System**

This section of the final report contains a high-level description of the OASYS software system, describes the design rationale, and supplies the design history necessary to provide context for that rationale. A detailed description of system operations is available in the User's Manual.

### **6.1 System Overview**

OASYS consists of two related software packages which are meant to be used together to conduct operability assessments of software/hardware/human systems. The data gathered from the OASYS system can be used to assess system usability, manning, and skill level requirements as well as overall system performance under the assumption that the hardware is performing up to specifications. (The OASYS system is not intended to assess hardware reliability issues, though it can be used to assess the operability impact of equipment failures.) In the following sections the term *target system* is used to describe the software and hardware system for which potential operability is being assessed. The term *user* refers to a user of OASYS, and the term *operator* refers to a user of the target system.

The two software packages are the OASYS Task Analysis System (OASYS-TAS), used for timing and load analysis, and the OASYS Experiment System (OASYS-ES), used for human-in-the-loop and model-in-the-loop experiments.

OASYS-TAS is a scenario-based task-load analysis system. It is a modeling tool that allows task performance in specific operating scenarios to be examined in detail. It allows the user to input a hierarchical description of the task breakdown for the activities being assessed. At each level of the hierarchy, the tasks can be arranged into a task-flow network that captures the order and time interdependencies among them. The user can also include conditional branches and other control nodes (logic rules) to indicate that the execution of a task sequence must wait for some particular event or that the execution of the task causes some event to occur.

Individual tasks in the hierarchy can be annotated with data concerning each task. In particular, the user may specify the performer of the task, the time required for the task, and the priority of the task. These attributes and the sequencing information allow the task decomposition to serve as a basis for examining particular operating scenarios.

OASYS-ES is a human-and-model-in-the-loop operability testbed. It facilitates the construction of part-task or whole-task simulations of a target system and then allows experiments to be conducted on that simulation. The operator roles in the system may be played by human beings or by computerized human performance models. The system can be run in different configurations and in different simulated scenarios. The user can determine what data are to be collected during the experiment and how the experiment will be set up in terms of trials, subjects, and target system configurations.

OASYS was originally envisioned as a single integrated system that supported all aspects of operability assessment, using a single object-oriented data base. In February of 1995 the OASYS effort was refocused on support of human-and-model in the loop experiments. At the same time further work on the OASYS data base was discontinued and several features were dropped from consideration. As a result of this refocusing, the OASYS system was divided into the two subsystems described above. By de-emphasizing the data base, we traded the ability to link different relevant data from different parts of OASYS for increased system openness. The experimental system was "opened up" and reconceived as a set of tools which could be used together and in combination with COTS and custom software to facilitate operability experiments in a wider variety of technical circumstances. For instance, if OASYS were to be used early in the design of a new system, all OASYS components might be used to run early qualitative and quantitative design-viability checks. If OASYS were brought into an ongoing design effort late in the game (as operability considerations so often are), then some OASYS experiment tools might be used in conjunction with existing simulations or physical prototypes of operator stations.

The OASYS software systems are intended for a variety of users who will work together in the analysis of system operability. The OASYS-TAS tool and many aspects of the OASYS-ES experimental system, in particular experiment design, script writing, and interface specification (described below), are designed for users with no programming knowledge. Building the behavior of equipment simulations and creating new simulated objects and GUI gadgets requires the participation of a Lisp programmer.

## **6.2 Documents**

An important concept in the design of OASYS was the ability to link paragraphs in documents to each other and to other objects in OASYS such as GUI gadgets, simulations, and task descriptions. In order to facilitate maintaining these links as the documents are edited, it was decided to implement a text and document editing system as

part of the OASYS implementation. The format for saving documents with their links was based on the hypertext concept .

A preliminary version of a text editor with linking capability was implemented, but the cost of extending this version to a complete document editor would have taken resources away from more unique features of the system.

The document handling approach was reevaluated, and we recommended that document generation and editing should be done outside of OASYS using one of the many excellent document systems available commercially. Documents would be imported into OASYS in digital format (such as Rich Text Format (RTF) or Continuous Acquisition and Lifecycle Support (CALs) format). Within OASYS, paragraphs could be linked to OASYS objects including other documents. When a new version of a document was imported, links to paragraphs that were not changed would be propagated to the new document. Links to changed paragraphs would be listed and the user would be required to reconnect them to appropriate paragraphs.

Work had begun on importing the two formats when the effort was refocused. As part of the change in emphasis, the document tools in OASYS were de-emphasized and eventually taken out of the system.

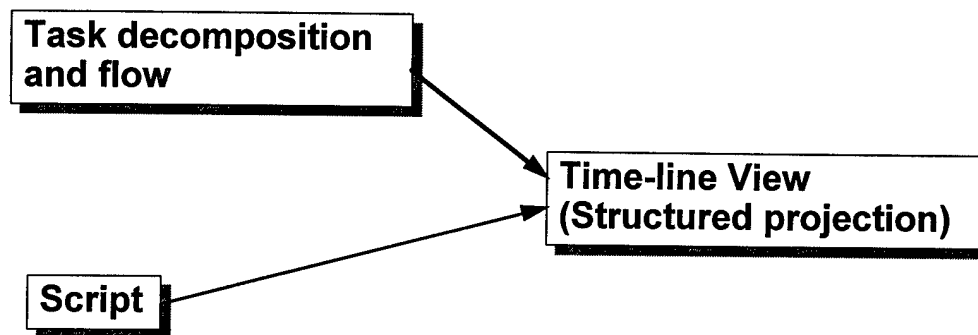
In retrospect it is clear that the idea of implementing a new document system was unrealistic and that the only practical approach was to interface to a COTS systems using one of the standard formats such as RTF.

### **6.3 Task Analysis, Timing, and Load**

The OASYS-TAS task timing and load system is designed to support the methodology of scenario-coupled operability assessment. This means that it is focused on exploring the operability ramifications of individual, specific situations rather than on the amalgamation of statistics over a large number of cases. There were two reasons for this choice. First, stochastically based task flow analysis tools already exist and are in wide use (e. g., Saint and Microsaint). Second, we felt strongly that many operability problems, all of which involve a human component, are more readily understood and corrected in specific contexts rather than in terms of average times and error rates.

Support of scenario-coupled assessment is accomplished through three components of the timing and load system—the task decomposition tool, the script editor, and the timeline viewer, as illustrated in Figure 2. The task decomposition tool will be familiar to users of stochastic task analysis tools such as Microsaint or to users of project planning tools such as Microsoft Project. As described in the system overview, it is used to input,

edit, and view a hierarchical description of the tasks and subtasks for which operability is to be assessed. It allows each task to be annotated and it allows the subtasks of each task to be arranged in a flow chart that describes their order of execution. Also included in these flow charts are "semaphores" representing the decisions made during task execution. These semaphores are used to represent the need for a task to synchronize with other tasks that might be executing concurrently or to wait for events in the external world.



**Figure 2. Task Analysis Tools**

In order to represent and manipulate specific scenarios, OASYS-TAS incorporates an element not found in other task analysis tools—a script that provides the specifics of each scenario to be examined. These scripts are created by the user through the script editor and contain the relevant events and conditions that would occur in a particular scenario. We envision that an analyst might write several scripts when assessing the operability of a proposed system in order to represent various possible operating conditions such as heavy load, equipment failures, exceptional circumstances, normal load, etc.

OASYS-TAS was designed to support the analysis of short, intense periods of activity as well as longer periods in which idle time and delays may occur. For short periods when many activities occur, the timeline view may be displayed with each task drawn proportionate to its duration. For analysis of long periods of low activity, a compressed summary view shows only task starts and stops so that long periods of elapsed time may be collapsed onto a small display.

The initial vision and design concept for OASYS specified a single unified system that supported all aspects of operability analysis. Operability analysis in practice is usually not a well-defined, integrated, and well organized process, however. Instead, it often involves piecemeal analysis and testing of different aspects of the system at

different phases of design and development. For this reason, the final OASYS product was eventually reconceived as a set of tools, rather than as a single monolithic tool.

In several ways, the timing and load tool was a casualty of this initial "unified system" approach to OASYS. OASYS-TAS provides capabilities that are independent from those of the other experiment-oriented components of OASYS, and it was the first component of OASYS to be developed. It should have been delivered and tested early, as a separate component, rather than being tied to more-complex OASYS capabilities that were not developed until later in the project. Earlier delivery and testing of a self-contained timing and load analysis tool would have allowed for more iterations in the design and could have led to a better final product. Depending on user feedback, other features could have been added. For example, the usefulness of allowing the user to specify task preemption based on prioritization (if two tasks conflict for resources, the user can specify which should be done first) was discussed, but project resources were not available to add this feature.

The focus on "vertical" integration of the timing and load tool with the other OASYS system components took precedence over providing "lateral" interoperability with other related tools that might have proved more useful. For example, the possibility of making the timing and load tool interoperable with a stochastic task analysis tool such as Microsaint was discussed, but never implemented. Interoperability with other complementary task-analysis tools could have increased the usefulness of OASYS-TAS from the user's perspective.

The timing and load tool (as well as the rest of the OASYS tool set) also suffered from a lack of unification and consistency in the interface design. Several different software developers were involved in different aspects of the tool, and each implemented a slightly different interface style. Each style had its own advantages, but the inconsistencies in the interface were disconcerting and annoying to the user, and proved to be time consuming to correct late in the development process. In retrospect, it is clear that a unified style for the interface look and feel should have been imposed early in development.

#### **6.4 Experiment System**

The OASYS-ES experiment system can be conceived as a collection of cooperating tools that can be used together, as illustrated Figure 3. OASYS-ES supports the development of a GUI linked to a simulation of the performance of the equipment in the target system. This dynamic GUI may be used in demonstrations or in systematic

experiments designed to collect performance data. GUIs may be operated by test subjects or by human-performance models.

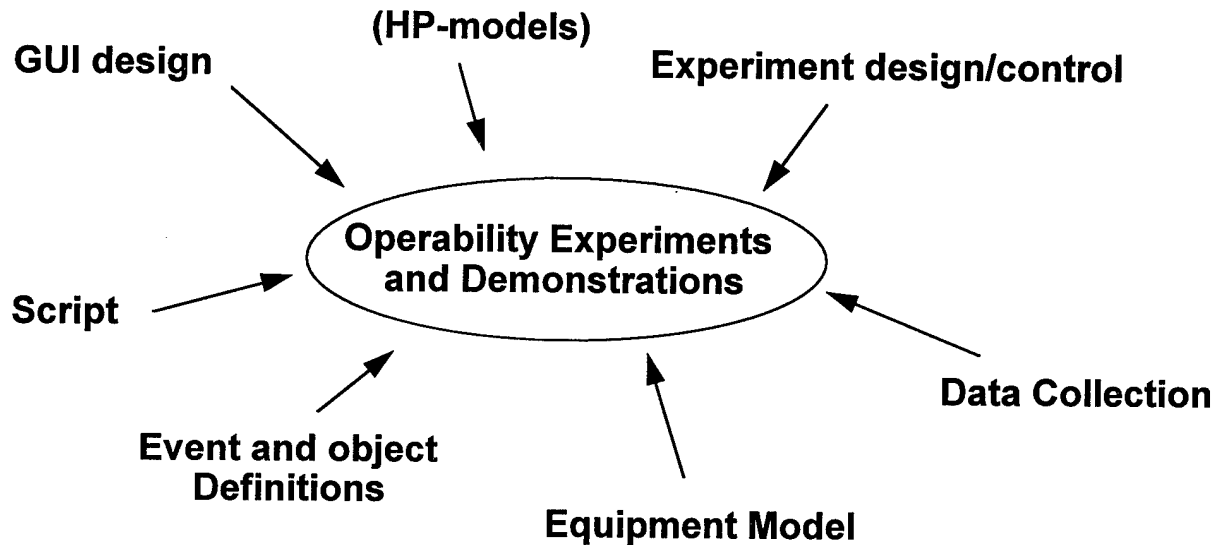


Figure 3. OASYS-ES Experiment System

#### 6.4.1 Interface Design

The OASYS user can create the interfaces through which human operators interact with the simulated system. OASYS provides a panel editor for creating the "windows" that will comprise the interface(s) to the target system. These windows can be populated with gadgets from a palette using drag and drop interaction. These gadgets may be moved, resized, and edited for color, text font, text size, etc. Gadgets are linked to event translators that accept input from or provide input to the gadgets.

The GUI-building capability implemented in OASYS supports two levels of users—operability analysts without programming experience and Lisp programmers. Nonprogrammers may create interface designs by selecting gadgets from the palette, positioning them on screens, and customizing their appearance. A "smart workstation" capability helps the user to create the Event Translators needed to make GUI gadgets function dynamically. The smart workstation is based on the assumption that a gadget will be used in a typical way, and automatically generates the Event Translators needed to support this typical use. New gadgets may be created and added to the palette, but this requires Lisp programming (see the Programmer's Manual).

A decision was made early in OASYS design not to use one of the many available COTS GUI-building tools, but instead to devote effort to the development of a Lisp-



based tool for creating display gadgets and using them to build GUIs. There were several reasons for this decision. At the time of the decision, most GUI-building tools allowed the user to manipulate a graphic representation of the display, then, in a separate step, generated the C code needed to create that display. Whenever changes were made to the display design, this process had to be repeated. The few GUI building packages that allowed direct manipulation of the display without a separate code-generation step were extremely expensive, and even these packages had little capability for interactive data interchange with an external simulation.

Our original design for OASYS envisioned capabilities that would have been difficult to implement with the COTS GUI-building tools available at the time. In the original design, all objects in the system were stored in one data base and the user was allowed to create hypertext links between objects in that database. GUI gadgets were objects in this data base, allowing the user, for example, to create a link between a gadget on the GUI and a requirements-specification document also stored in the data base. The user could then view the GUI that satisfied a requirement by clicking on the relevant paragraph of text to bring up the GUI. Neither this hypertext-linked data base nor the smart workstations described above would have been feasible with COTS GUI builders.

In retrospect, the decision not to use a COTS GUI builder seems questionable. We succumbed to the allure of tackling a solvable problem, rather than really considering whether the problem needed to be solved. Although the panel editor (called "Mirage") was successfully implemented, it consumed considerable resources that might have been used elsewhere. Building Mirage was something that *could* be done, but not, perhaps, something that *should* have been done. Also, in the time since the decision was made, COTS GUI builders have improved considerably in capability and have decreased in price.

#### **6.4.2 Simulation Design**

OASYS supports the design and construction of system behavior for the target system. Our original goal was to build a set of tools that would allow a non-programmer to construct complex system behavior. We now believe this objective is, in some sense, impossible by definition. At its core, it is the nature of the programmer's expertise to be able to construct a set of procedures, objects, heuristics and algorithms which, when executed, will produce a desired behavior. Trying to enable non-programmers to accomplish this task often results in the creation of a new (perhaps graphical) programming language. To the extent that this language is well designed and tailored to its task, it may be easier to learn than the original language. Although its power is

limited, it may be fundamentally acceptable to a wider audience. In no case, however, is a programming task possible without programming. We eventually came to believe that the best that can be done is to create an environment that is easy to use and to strike a balance between the range of equipment behavior that can be simulated and the expertise required to realize this behavior. Much of our early design effort was exhausted discovering this truth.

As it stands now, the OASYS simulation design system is a set of tools which, to a degree, isolates a programmer from the need for extensive knowledge of the underlying Lisp language in which the simulator is implemented. It allows simulations of moderate complexity to be created entirely through a series of graphical interfaces. The resulting simulations can be tailored by non-programmers and will be intelligible enough to them to allow meaningful data collection to be specified during experiments.

Building a simulation model in OASYS is a process of building components. First, the modeler defines the kinds of objects and events that will exist in the simulation, e.g., in an air traffic control simulation, planes may be one type of object, and planes crossing an airspace border may be one type of event. Objects and events are created using a type editor that allows the modeler to create a hierarchical representation of types. Planes, for example, might be one type of vehicle. For each type, the modeler specifies the attributes relevant to that type of object, e.g., planes might have altitudes associated with them. Properties are inherited according to the hierarchical type structure specified by the modeler. For example, if all types of vehicles have a position associated with them, then planes, as a type of vehicle, will have a position attribute.

As part of simulation design, the modeler specifies the operator stations that will exist when the simulation is run. Each operator station defines a position to be manned by an operator or a human performance model when the simulation is run, and consists of the set of GUI panels relevant to that operator. The physical location of the operator station, defined by the specific workstation on which the GUI panels will be displayed, is defined as part of the experiment set up before a simulation is run.

In order to run a simulation in OASYS, a script must be defined. This script specifies the specific objects that are to exist in the simulation, e.g., five airplanes of the same type, and the initial attributes of these objects, e.g., the initial position of each plane. Scripts also specify any external events that arise outside the simulation but have an effect on simulated objects and events. One simulation model may have many different scripts associated with it.

Event translators are the basic mechanism for making things happen in the simulation. Event translators are rules that express the consequences of events occurring. Each event translator has three parts: 1) a specification of the kind of input event that triggers the event translator; 2) a specification of any attributes of an object that change as a result of the occurrence of the event; and 3) a specification of the output event(s) generated by the event, if any. Output events may have a time delay associated with them.

The major lesson we learned from building the OASYS simulation-construction tools, as discussed above, is that building a powerful simulation language and building a language that can be used by nonprogrammers are inherently contradictory goals. In retrospect, we should have provided a procedure-based language for non-programmers that could be used to specify linear sequences of simulated events, and a set of tools for programmers that could be used to create more powerful and complex simulations.

#### **6.4.3 Experiment Design**

OASYS-ES allows the user to design a human-in-the-loop (or model-in-the-loop) experiment by specifying a set of experiment design parameters. Essentially, an experiment is designed to systematically vary some factor of interest for operability assessment, while controlling for extraneous sources of variability that are not of interest. The OASYS experiment-design parameters reflect the major factors likely to be of interest in an operability experiment: the design of the GUI, the performance of the hardware/software components of the target system, the "situation" (external events) as reflected in the script, and the characteristics (e.g., skills or training) of the subjects themselves.

An OASYS Experiment Design consists of a sequential series of experiment trials. To initiate a design, the user specifies the number of trials in the experiment, the number of subjects, and the number of operator positions/workstations to be used. For each experiment trial, the user then supplies the identifying number for the subject(s) who will participate in that trial, the assignment of subject(s) to operator stations or position (with associated GUI panels), the script to be used, and the system model that will be used to simulate the hardware/software behavior of the target system. For multi-person teams of subjects, there is also a group identifier associated with each team.

The experiment design also specifies the "mode" in which each trial in the experiment will be run. Trials in auto mode are used for both training and data collection trials. Once an auto trial is initiated, the user (experimenter) has no control over the trial other

than the ability to interrupt it. Manual trials are used for debugging purposes, and allow the user to intervene (pause, resume, etc.) during the trial.

The physical workstations to be used for each operator station are specified as part of the Experiment Design. This is done in two stages. The Experiment Design Window specifies operator stations by ID code, e.g., "A," "B," etc. As one of the last steps taken before selected trials are run, the experimenter specifies which physical workstations (by system name) are to be used to create each operator station. These assignments may be used to change the physical configuration of trials over the course of the experiment as needed, e.g., trials may be run at several different locations. The assignment of operator positions to workstations for trials not yet run may be changed as needed during the experiment.

Experiment subjects may be either human beings or human performance models such as those built with the Operator Model Architecture (OMAR) program. For experiments involving multiple operators, it is possible to designate some operator positions to be filled by human subjects while others are filled by models. The designation of humans or models as subjects may be changed over the course of the experiment by changing experiment design information for trials that have not yet been run.

If data are to be recorded for trials in an experiment, the user must specify the events (from the simulation) to be recorded. This is done by selecting event types from the OASYS Type Hierarchy. When the simulation is run during an experiment trial, OASYS records the time and the parameters of each event for the event types selected, creating a data file for later analysis.

During OASYS design, we considered providing "smart experiment design" capabilities. This might have allowed the user, for example, to specify the number of independent variables in the experiment, their levels, and whether they were to be within-subjects or between-subjects variables, and to specify the number of replications. The system would then have generated a suggested experiment design. We eventually abandoned this idea because of the large variety of experiment designs potentially relevant for operability experiments and the number and complexity of the factors that affect the "best" design. We still believe this decision to be the correct one. OASYS as implemented does not tell the user how to design an experiment, but instead provides enough flexibility for the user to set up a wide variety of experiment designs appropriate in different situations.

We also faced a tradeoff in design between supporting experiments involving a few long trials and those involving many short trials. Some operability experiments, for

example, might involve only a few "scenarios," each of which lasts for several hours. Others might involve hundreds of repetitions of a task which takes only a few seconds. The capabilities needed for long-trial experiments may be useless for short trial experiments, and vice versa. For example, the ability to pause and resume in the middle of a trial may be needed for long trials, while the ability to automatically sequence many trials to run without experimenter intervention may be needed for short trials. Again, our design is a compromise which supports both types of experiments. The user (experimenter) may select a large number of trials to be run automatically for a subject, or may select only one trial to run. Trials may be run in an auto mode, in which the experimenter has no control over the trial once it starts, or in a manual mode, in which the trial may be stopped and resumed at any time.

The experiment-design portion of OASYS benefited from Air Force involvement through an iterative design process. We designed a series of storyboard sketches for the experiment-design interface for Air Force review, and were able to benefit from comments and suggestions based on these storyboards. This iterative design process would probably have been advantageous for other OASYS components, but it was not implemented until relatively late in the project. Also, the OASYS experiment-design capabilities had an obvious application, and our Air Force reviewers were able to draw directly on their own experience to generate useful comments. Some of the more abstract OASYS elements (e.g., the Event Translators) were more difficult to review constructively during design.

Overall, we feel that the design of the OASYS experiment-design component was relatively successful. This success was due to a series of reasonable compromises between flexibility and maximal support, an iterative design process with the Air Force, the benefits of prior experience on the project, and the concrete and easily understandable nature of the component's capabilities.

#### **6.4.4 Experiment Execution**

When an OASYS simulation starts, objects are created based on the script that has been selected. An event queue is created and populated by the external events specified in the script. As the simulation runs, events may be added to this event queue. The simulation runs via an execution loop that takes the next event, waits for the time at which the event is scheduled to occur, and causes that event to occur. The simulation then checks to see if any event translators are waiting for the event to occur and executes the rules associated with these event translators, including the possible creation of new

events. When the operator interacts with GUI panels, new events are generated that may be responded to by event translators.

The speed with which complex simulations would execute during experiments was a source of considerable concern during OASYS design. Simulations driving human-in-the-loop experiments must run quickly enough to support human-machine interactions on a time scale equivalent to that of the target system. Because execution speed is a function of many complex factors, it was difficult to predict how quickly an OASYS simulation would execute while the system was being designed. When OASYS was completed and tested, execution speed did not prove to be a problem for the limited demonstrations conducted during the program. It is not clear that this would be the case for more complex systems, however.

OASYS is designed to support multiple distributed operator stations, and it offers the possibility of conducting distributed experiments at remote physical locations. We explored the use of the Internet to conduct such experiments, with an OASYS simulation driving workstations at several geographically distant locations. Conducting a distributed experiment over long distances would require the existence of a dedicated communications channel, however. Because the Internet does not supply dedicated channels, there is always a possibility that unpredictable delays in message transmission could cause unacceptable delays in experiment execution.

#### **6.4.5 Data Collection and Analysis**

For all experiment trials in which data collection is chosen as an option, OASYS records data on user-selected event types, including the time of the event and any parameters associated with the event. The user selects event types to be recorded from a list of all of the event types in the simulation. The user may choose to have all data for a series of trials recorded in one data file, or to create a separate data file for each trial.

An early decision was made to produce OASYS experiment data in a form that was accessible to a COTS statistical analysis package, rather than adding data-analysis capabilities to OASYS. OASYS currently produces data files that may be analyzed with the Statistical Analysis System (SAS), probably the most widely used data analysis package in the behavioral sciences. OASYS data files may also be converted into a spreadsheet format and analyzed using Excel or other spreadsheet packages.

An initial goal for OASYS was that it produce data on manning and skill-level requirements for the target system. We investigated the possibility of having OASYS produce manning-requirements data files that were directly usable by several different

military manpower-planning models. Unfortunately, the issue of what type of data from OASYS experiments would be relevant for manpower planning and skill-level requirements became confounded with the issue of how OASYS data collection should be designed. Final design and development of the OASYS data collection capability was postponed contingent on investigating the type of data and data format needed by various manpower planning models. Eventually, we concluded that most data from OASYS experiments would not be directly useable by existing manpower and skill-level planning models, that the major OASYS data relevant to such models would be task-timing data, and that these data would be usable in almost any format. Our initial decision to produce data files readable by COTS analysis package was the correct one, and OASYS data collection could have been implemented much earlier if we had realized that experiment data files need not be tailored for existing staffing and skill-level requirements models.

#### **6.4.6 Human Performance Model Interface**

It was central to the overall concept for OASYS that it accommodate model-in-the-loop as well as human-in-the-loop operability testing and experimentation. The completed OASYS-ES tool features complete interchangeability between human subjects and human performance models developed using the Operator Model Architecture (OMAR) toolset. In setting up an experiment, operator stations may be assigned to physical workstations for use by human operators, or to human performance models. During an experiment trial, data on operator-system interactions are collected for both human operators and models as specified in the experiment design. In a multi-operator experiment, some operator positions may be filled by human operators at the same time that other positions are filled by models, with identical performance data collected for all positions.

We expended considerable OASYS design time in attempting to develop a detailed specification of the interface between OASYS simulations and human performance models developed using OMAR. Developing a specification, in the abstract, for integrating two systems that were "moving targets" proved to be extremely difficult. Ultimately, we found that our best strategy was to work from a concrete example and to develop a design by tackling a real problem. We built a demonstration (described below) in which human operators and OMAR models interchangeably control a simplified air traffic control system. In the course of implementing this demonstration, we developed a workable design for the OASYS-OMAR interface.

Eventually, we developed an event-based interface for linking OMAR human performance models to the OASYS simulation. The human performance model is

responsible for specifying to OASYS which events it wants to "hear." When the OASYS simulation executes, OASYS broadcasts these selected events to the model, including any parameters associated with each event. As part of the demonstration, we provided a set of Lisp-based human performance models that interacted with an OASYS simulation by "listening" for events and injecting new events back into the OASYS event queue.

#### **6.4.7 Demonstration**

As part of the final demonstration of OASYS, we built a simplified air traffic control (ATC) simulation for a team of four operators controlling adjacent sectors of the airspace. Operators exchanged structured messages in order to transfer aircraft between sectors, and were able to accept or deny these transfer requests as a function of their workload. A script controlled the number, route, and speed of aircraft, subject to the commands of the ATC operators. Two scripts were implemented to create different levels of workload for the operators by varying the number of planes in the airspace. An OMAR model was built to handle the ATC task, and this model was able to fill any (or all) of the operator positions, with human operators filling the remaining positions. Comparative data were collected for all four positions, with three positions filled by models and one by a human operator. Measures of performance included the number of times that aircraft were delayed waiting for a clearance, the mean length of the delays, and the mean time between receiving a clearance and sector crossing. The data file produced by OASYS was transferred to an Excel spreadsheet for analysis.

This demonstration illustrated many of the key features of the OASYS experiment system: the ability to quickly create a simulation of a target system, the ability to easily create dynamic GUI prototypes, the ability to simulate multiple operator stations for a multi-person task, the ability to interchangeably use human operators and human performance models in an experiment, the ability to specify performance data be collected during the experiment for both humans and models, and the ability to create an experiment data file that can be analyzed with COTS software.



## **7. OASYS Software Testing**

Testing of the OASYS systems (i.e., the Task Analysis System (OASYS-TAS) and the Experiment System (OASYS-ES)) specifically addressed the graphical interface and the individual system commands used as parts of procedures. To ensure that the systems looked and performed as required, they were tested against their description in the OASYS User's Manual to verify their appearance and behavior. The manual serves as a top-level specification of how the graphical interface should look to the user and how the user performs tasks using the system commands. For each system, the User's Manual contains descriptions of all of the system windows and step-by-step procedures to perform a task analysis on a target system or to design and execute an operability experiment using OASYS.

### **7.1. Test Coverage**

Testing of the graphical interface involved comparing the on-line screens to the descriptions of the windows for accuracy and completeness. Did the windows appear as described? Did they contain the menu items, buttons, cells, panes, etc. as described? Were the names of windows, dialogs, menus, buttons, and labels correct?

In testing the system behavior, each procedure was performed as described in the manual to see if the system produced the desired result. The testing verified that the steps were complete and in the correct order, and checked any aspect of the graphical interface that resulted or changed as a result of following the procedure. For example, the manual describes how to edit the timing and load values of a task object (the TAS System, pg. 36-37). In testing this behavior, the test was passed if the dialog box for changing the values was invoked by the indicated action, if the dialog box contained all the fields described, and if values could be entered successfully into the fields.

The testing effort did not extend to the integration of the system components. For example, when running a task analysis, the time to execute a task is a task attribute that is set by following the procedure to edit the attributes of a task object. Both the procedure for editing a task object and the procedure for running a task analysis were tested separately. We believe, based on informal observation, that task attributes were applied correctly when running a task analysis, but this was not formally tested.

## **7.2 Test Process**

The windows examined, the procedures tested, and the order of testing was determined by the content and structure of the User's Manual. The manual was followed page by page. When sections detailed the layout and components of a window, that window was activated and examined for compatibility with the manual description. When sections outlined step-by-step procedures for performing a function, that procedure was followed as described and the results were examined for both visual and functional correctness.

The results of the testing are presented in the Software Test Report. The report itemizes all the windows and functionality tested and gives the result of testing each item.

## **8. Lessons Learned**

OASYS design attempted to use a rapid prototyping approach in which system features are prototyped, demonstrated, and made available for hands-on testing by a potential user community. User feedback is then incorporated in the design to produce a system that meets user needs and is easy to use by the target community. Application of this design approach was only partially successful for OASYS, and the experience and difficulties encountered provide some useful lessons for future projects.

Although there was a significant effort to get a sample user community to review and test the OASYS prototypes (see Section 4.0), there was still not enough detailed timely feedback to support the rapid prototype design approach. With the possible exception of the AFOTEC representatives, no one on the RRB would actually use a system like OASYS. The human factors experts from AL were potential users, but they represented only a small specialized group of users and they did not have the time to try to use OASYS for a realistic task. For a rapid prototype design effort to work well there must be one or two real users working almost full time applying the prototypes to a real task and interacting with the design team on a daily basis.

The distribution of time between design and implementation on the OASYS development was more representative of a conventional design process than a prototype design effort. When a rapid prototype design process is used, the design phase must allow time for several cycles of prototype and test use of the major system features, but a major part of the final prototypes are reusable in the implementation phase. The time allocated for design should, therefore, be significantly longer than the implementation, the reverse of the normal development cycle.

A rapid prototype design approach does not reduce the importance of having a clear definition of the system requirements and concept of operation and a thorough understanding of the specific user community for the system. System Requirements and Concept of Operation Documents should be prepared and reviewed to provide a guide for prototype efforts.

A rapid prototype design approach does not reduce the importance of having agreement on the look and feel rules for the HCI for designers (prototypers) of all subsystems or tools to follow. Look and feel rules should be documented and agreed to before prototyping starts.

When the system consists of a set of tools some of which have functions similar to commercial tools such as document editors or GUI builders, the effort should be strongly biased toward using the commercial tools and building interfaces and shells to integrate them rather than building the tools themselves. This bias should be considered even when system requirements must be relaxed or modified to allow incorporation of COTS packages.

Another lesson learned from OASYS is the difficulty of building a set of tools that allow a non-programmer to construct complex system behavior. Over the course of the project, we came to believe that this objective is, in some sense, impossible by definition. It is the nature of programming to construct a set of procedures, objects, heuristics and algorithms which, when executed, will produce a desired behavior. Tools can make this process easier, but they cannot replace the perspective and the design skills needed create a complex simulation. We eventually came to believe that the best that can be done is to create an environment that is easy to use and to strike a balance between the range of equipment behavior that can be simulated and the expertise required to realize this behavior.

## **9. In Retrospect**

The OASYS system was conceived as a single tool to satisfy the needs of a wide range of users including system designers, human performance modelers, human factors experts, system test planners, and others involved in system operability. It was intended as a powerful experiment building tool that would eliminate the need for programmers when doing operability experiments. In retrospect both of these goals seem too ambitious for the current state of the art.

Among the possible users that reviewed the system design for OASYS there seemed to be very little overlap in the features they wanted to add or change. System designers

and human factors experts both wanted the ability to run experiments, but the type of experiments of interest were different. Human factors people wanted short experiments that could be repeated with groups of subjects to produce quantitative results. The system designers wanted longer total system experiments that may only produce subjective results and identify design weaknesses. Others were interested in process models based on the task decomposition rather than actual system models.

Although we made significant progress in reducing the need for computer programmers, it is clear that simulation of large system requires some form of formal programming language.

In the end we produced two systems, one that adds a significant new capability (scripted process models) to process modeling, and one that supports experiments with complex system models and allows interchangeable use of live humans or human performance models at the expense of requiring extensive programmer support.

## 10. References

- Fuld, R.B. (1993). The fiction of function allocation. *Ergonomics in Design*, January.
- Helander, M., Editor. (1988). *Handbook of Human-Computer Interaction*, Elsevier Science Publishers, Amsterdam, The Netherlands.
- Lysaght, R.J., Hill, S.G., Dick, A.O., Plamondon, B.D., Linton, P.M., Wierwille, W.W., Zakland, A.L., Bittner, A.C., Jr. and Wherry, R.J. (1989). *Operator Workload: Comprehensive Review and Evaluation of Operator Workload Methodologies*, Technical Report 851, United States Army Research Institute for the Behavioral and Social Sciences, Alexandria, VA.
- Meister, D. (1985). *Behavioral Analysis and Measurement Methods*. New York, NY: John Wiley & Sons.
- Preece, J. (1994). *Human-Computer Interaction*. Reading, MA: Addison Wesley
- Shneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Reading, MA: Addison Wesley.
- Singleton, W.T. (1994). From research to practice, *Ergonomics in Design*, July.
- Smith, S.L. and Mosier, J.N. (1986). *Guidelines for Designing User Interface Software*, Bedford, MA: MITRE Corporation.
- Young, M.J. (1993). Human Performance Models as Semi-Autonomous Agents, *Proceedings of the Fourth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*. Los Alamitos, CA: IEEE Computer Society Press.

### OASYS Documents

- 1 "Concept of Operations", CDRL A010, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, September 1993.
- 2 "System Requirements Specification" CDRL A008, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, November 1994.
- 3 "User Manual", CDRL A015, Contract F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, June, 1995.
- 4 "OASYS Programmers Manual", CDRL A014, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, June 1995.
- 5 "Installation Guide", OASYS Task Analysis System (OASYS-TAS), CDRL A004, Contract No. F33615-91-C-0012, prepared for Air Force Systems Command, Armstrong Laboratory, Wright Patterson AFB, OH, June 1995.